简单工厂：

通过参数来确定要创建的实现对象

通过反射来实现简单工厂的开闭原则

常用的源码：

Calendar，Jdbc

工厂方法：

具体的创建实例是在子类中创建

常用的源码：

Collection

UrlStreamhandlerfactory

抽象工厂：易于产品族编程，不易于产品结构编程

产品族：比如，美的空调，美的电视是“美的”的产品族

产品结构：比如，海尔空调，美的空调都是“空调”

源码：java.sql.connector; java.sql.statement

建造者:buider

简单时：工厂

复杂时：建造者

源码：StringBuilder；

**单例模式：singleton**

概念：私有构造器

线程安全

延迟加载：懒汉

序列化和反序列化安全

反射

实用技能：反编译，内存原理，多线程debug

单例与工厂

单例与享元

双重检查->私有构造器作用->静态块单例->饿汉式单例

安全问题：（8-6：源码分析）

反序列化和反射攻击：implement，serlise， readResolve；

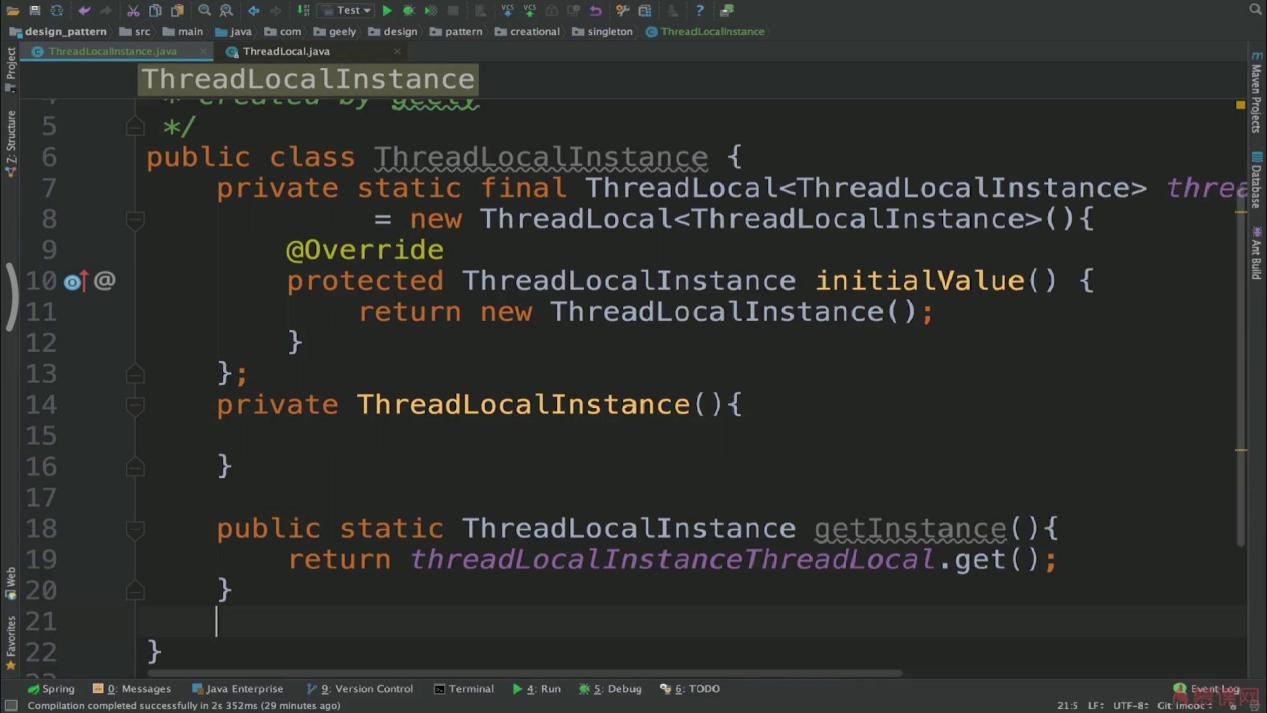
Enum枚举单例，原理源码解析以及反编译：

源码反编译工具：

<https://varaneckas.com/jad>

ContainerSingleton

基于ThreadLocal的“单例”：



为每个线程保留一个独立副本。

源码：Runtime；DeskTop;

Spring 单例：ErrorContext;

外观（门面facade）模式：

外观模式和中介者模式

外观模式和单例模式

外观模式和抽象模式

源码：jdbcUtils,Tomcat requestFacade

装饰者模式

servletRequest:

Mybaties

适配器模式（亡羊补牢adapt）：

类似电源适配器；使不同接口的类，有相同或者相似的结果时，使用；

考虑全局，复杂度；

复用现有的接口;

类适配器：

对象适配器：

例子：220V适配5V;

被适配的对象：220V；

目标target：5V;

Xmladapt;

享元模式(flyweight:轻量级)：

尽量少的创建对象

线程池

\*关注内/外部状态，线程安全

内部状态：在模式内部，不随外部变化而变化

\*复杂度增加

\*享元模式和代理模式

\*享元模式和单例模式

Example:

Employee

Manager

EmployeeFactory

Test

Interger.valueof();

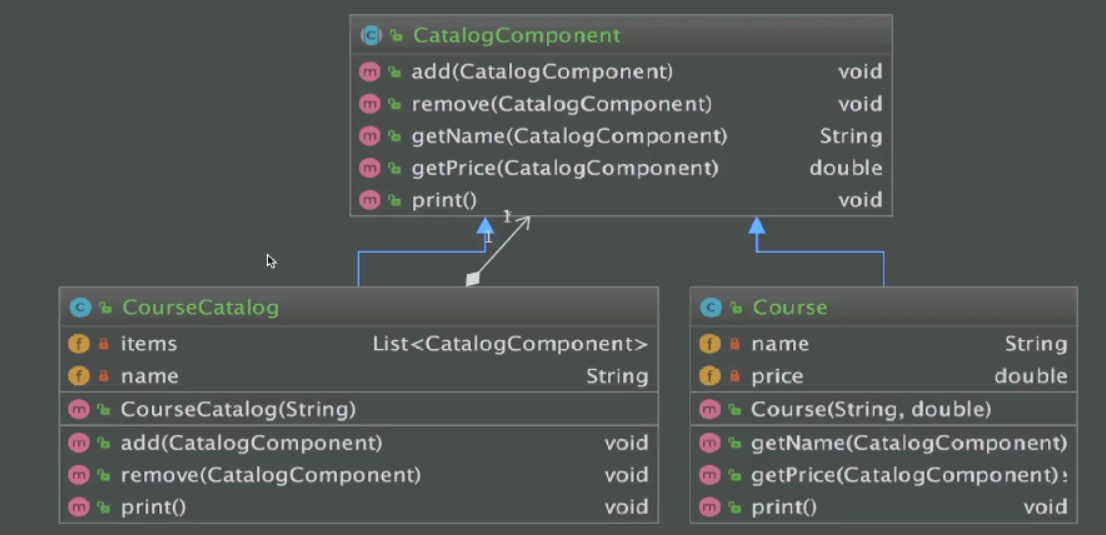
组合模式(节点

)：

处理树形结构

业务逻辑复杂不宜使用

\*组合模式和访问者模式



Java.awt

ArraryList addAll();

sqlNode

桥接模式（brige）；

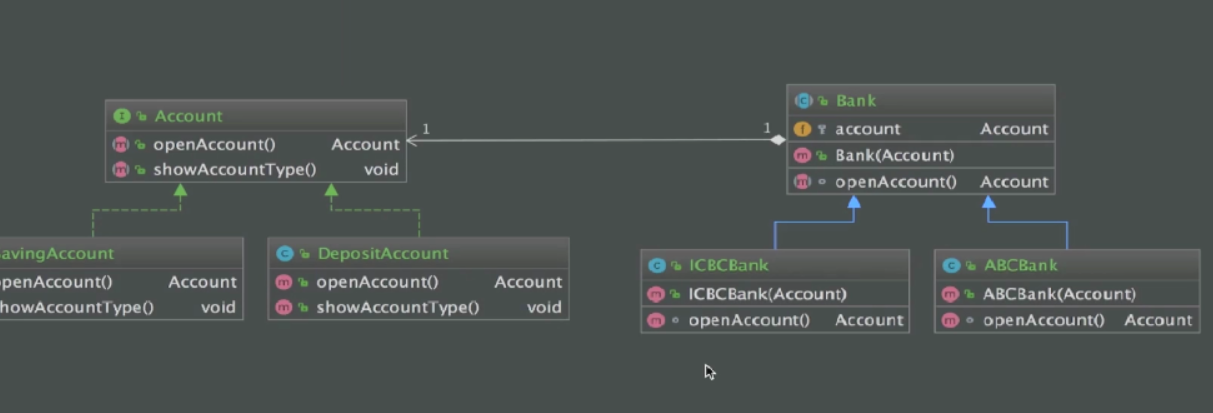
抽象不忍与实现部分分离，使他们独立的变化

通过组合的方式建立两个类之间的联系，而不是继承

抽象和具体实现之间增加更多的灵活性

桥接和组合模式：

桥接和适配器模式：

jdbc .drive

代理模式：

\*将代理对象与真实被调用的对象分里

\*降低耦合，扩展性好

\*保护目标对象

\*增强目标对象

静态代理

--

动态代理

--无法代理类，只能代理接口

CGlib

--类代理；注意final修饰符

Spring代理选择

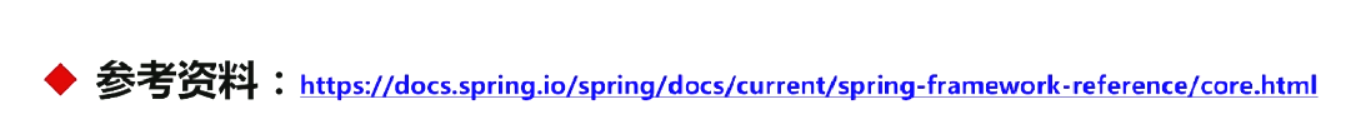
\*当Bean有实现接口时，Spring使用JDDK的东岱代理

\*当Bean没有实现接口时，Spring使用CGLib

\*可以强制使用Cglib

\*在Spring配置中加入





速度对比：

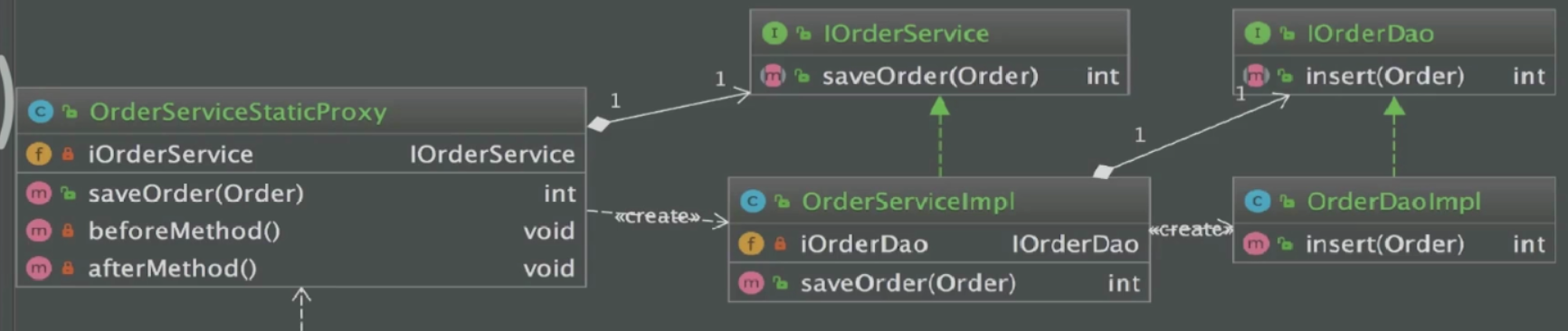
JDK>GC

代理模式和装饰者模式

代理模式和适配器模式

Codeing

--静态代理 proxy



--动态代理<AOP>

Reflect.Proxy

ProxyFactoryBean

JdkDynamicAopProxy

CgLibAopProxy

--CGlib代理

模板方法模式（behavioral）：

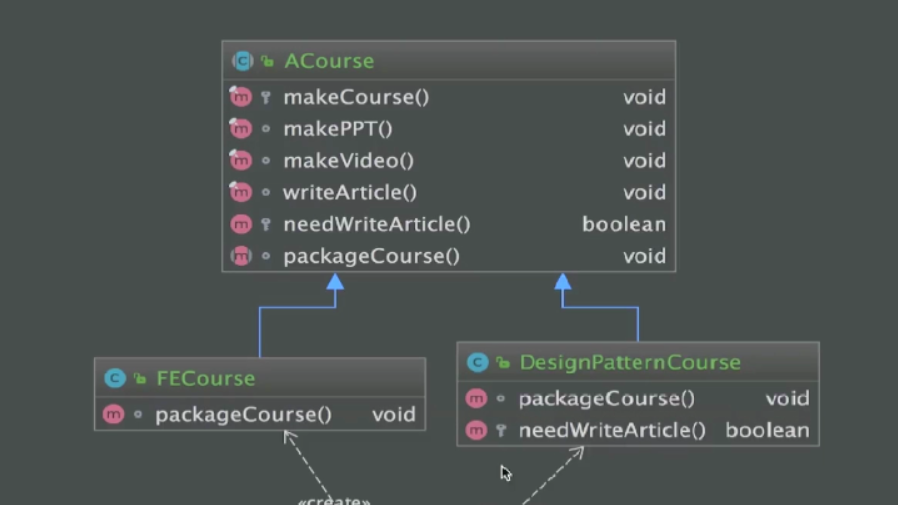
\*一次性实现一个算法不变的部分，并将可变的行为留给子类来实现

\*各子类的公共行为被提取出来集中在一个父类，避免代码重复

钩子方法：

模板和工厂方法

模板和策略模式（ifelse 使用）



AbstractList

AbstractSet

AbstractMap

HttpServlet